Centroid-based Clustering: Principles and Algorithms

Ahmed BADI
ahmedbadi905@gmail.com
linkedin.com/in/badi-ahmed

December 19, 2025

Abstract

Centroid-based clustering is one of the most widely used families of clustering algorithms in unsupervised learning. The central idea is simple but powerful: each cluster is represented by a prototype, called a centroid or medoid, and data points are assigned to the cluster with the nearest centroid. This article provides a focused introduction to centroid-based clustering. We explain the underlying optimization principles, the assumptions these methods make about the data, and the practical impact of those assumptions. We then study the most important algorithms in this family, including k-means, k-means++ initialization, and k-medoids, highlighting their strengths, limitations, and typical use cases. Through equations, figures, and intuitive explanations, this article aims to give a clear and accessible understanding of when and how to use centroid-based clustering effectively.

Keywords: Centroid-based Clustering, k-means, k-medoids, k-means++, Unsupervised Learning, Partitioning Methods.

1 Introduction

Clustering is about discovering groups of similar data points without relying on pre-existing labels. Among the many ways to define and detect such groups, centroid-based clustering takes a particularly intuitive approach: it assumes that each cluster can be summarized by a central prototype, and that points belong to the cluster whose prototype is closest to them [1], [2].

Imagine you want to segment customers into K market segments based on their spending patterns. Instead of describing each segment with complex rules, you could characterize it by a *typical* customer—a centroid vector that averages the behavior of all customers in that segment. New customers are then assigned to the segment whose typical customer is most similar to them. This is exactly the philosophy of centroid-based clustering.

Centroid-based methods are sometimes called **partitioning methods** because they explicitly partition the dataset into K non-overlapping clusters. The most famous algorithm in this family is k-means, but important variants such as k-medoids and k-means++ address some of its weaknesses.

In this article, we concentrate on:

- The general principle of centroid-based clustering.
- The k-means algorithm and its optimization objective.
- Initialization strategies, especially k-means++.
- The k-medoids variant, which is more robust to outliers.

2 Principle of Centroid-based Clustering

2.1 Clusters as Prototypes

Given a dataset $X = \{x_1, \ldots, x_n\}$ with $x_i \in \mathbb{R}^d$, centroid-based clustering assumes that:

- There are K clusters (chosen by the user).
- Each cluster k has a prototype vector $c_k \in \mathbb{R}^d$.
- Each point x_i is assigned to the closest centroid according to a distance measure $d(\cdot, \cdot)$.

In the simplest case, we use the squared Euclidean distance:

$$d(x_i, c_k) = \|x_i - c_k\|^2.$$

The goal is to find both:

- The partition of points into clusters $\{C_1, \ldots, C_K\}$.
- The centroid vectors $\{c_1, \ldots, c_K\}$.

such that the total within-cluster dispersion is minimized.

2.2 Objective Function

Formally, we can define binary assignment variables r_{ik}:

$$r_{ik} = \begin{cases} 1 & \text{if point } x_i \text{ is assigned to cluster } k, \\ 0 & \text{otherwise,} \end{cases}$$

with the constraint $\sum_{k=1}^{K} r_{ik} = 1$ for all i (hard clustering).

The generic centroid-based objective is:

$$J(\{r_{ik}\}, \{c_k\}) = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik}\, d(x_i, c_k), \tag{1}$$

and the algorithm tries to minimize J over both assignments and centroids.

Different choices of:

- distance $d(\cdot, \cdot)$,
- centroid definition (mean vs. medoid),
- optimization strategy,

lead to different algorithms within the centroid-based family.

2.3 Assumptions and Limitations

Centroid-based methods make implicit assumptions about the shape and distribution of clusters:

- Clusters are compact and roughly *globular* in the chosen feature space.
- Euclidean distance (or similar) is a meaningful measure of similarity.
- Each point belongs primarily to one cluster (hard assignments).

When these assumptions hold, centroid-based clustering is fast, interpretable, and effective. When they are violated—for example, when clusters are elongated manifolds, have very different densities, or when features are not scaled properly—results can be misleading.

3 k-means Clustering

3.1 Optimization Problem

k-means is the canonical centroid-based algorithm. It uses the squared Euclidean distance and defines centroids as means of assigned points [3]. The objective function is

$$J_{\text{k-means}} = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \, \|x_i - \mu_k\|^2, \tag{2}$$

where μ_k is the centroid of cluster k:

$$\mu_k = \frac{\sum_{i=1}^{n} r_{ik} x_i}{\sum_{i=1}^{n} r_{ik}}. \tag{3}$$

Minimizing $J_{\text{k-means}}$ exactly is NP-hard in general, but an efficient iterative algorithm finds a local minimum.

3.2 The k-means Algorithm

Algorithm 1 k-means Clustering

1: Choose number of clusters K.
2: Initialize centroids $\mu_1, \ldots, \mu_K$ (e.g. randomly or with k-means++).
3: **repeat**
4: **Assignment step:**
5: For each point x_i, assign it to its nearest centroid:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \|x_i - \mu_j\|^2, \\ 0 & \text{otherwise.} \end{cases}$$

6: **Update step:**
7: For each cluster k, recompute centroid as the mean of assigned points:

$$\mu_k = \frac{\sum_{i=1}^{n} r_{ik} x_i}{\sum_{i=1}^{n} r_{ik}}.$$

8: **until** assignments stop changing or maximum iterations reached

This algorithm is a special case of *coordinate descent*:

- Fix centroids, optimize assignments (assignment step).
- Fix assignments, optimize centroids (update step).

Each step cannot increase the objective $J_{\text{k-means}}$, so the algorithm converges to a local minimum after a finite number of iterations [1].

3.3 Geometric Interpretation

Geometrically, k-means partitions the space into K **Voronoi cells** determined by the centroids:

$$\text{Cluster } k = \{x : \|x - \mu_k\| \leq \|x - \mu_j\| \quad \forall j\}.$$

Decision boundaries are linear (hyperplanes halfway between centroids), which naturally leads to convex, roughly spherical clusters in Euclidean space.

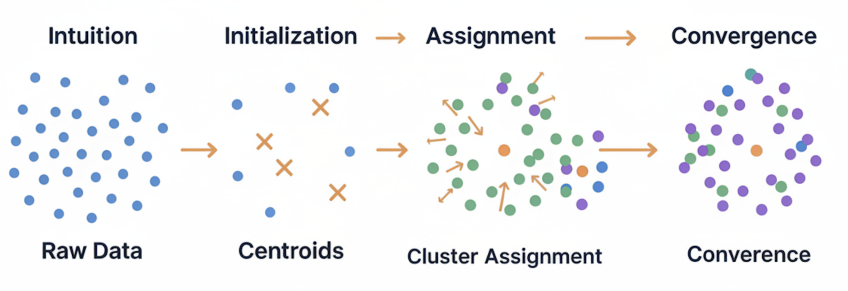

Figure 1: Illustration of k-means clustering: initial random centroids, assignment of points, and centroid updates over iterations.

3.4 Strengths and Weaknesses

Advantages:

- Computationally efficient and scalable to large datasets.
- Simple to implement and widely available in libraries.
- Works well when clusters are well-separated and compact.

Limitations:

- Requires pre-specifying K.
- Sensitive to initial centroids; different runs may yield different results.
- Sensitive to outliers, which can drag centroids away from dense regions.
- Assumes spherical clusters with similar sizes and densities.

These limitations motivate improved initialization strategies and robust variants such as k-medoids.

4 Initialization: k-means++

A major practical issue with k-means is that poor initialization can lead to bad local minima (suboptimal clusterings) [1]. The **k-means++** algorithm proposes a smarter way to choose initial centroids, dramatically improving both speed of convergence and solution quality.

4.1 k-means++ Strategy

The idea is to spread initial centroids out in regions of high density:

1. Choose the first centroid μ_1 uniformly at random from the data points.
2. For each remaining point x_i, compute its distance $D(x_i)$ to the nearest already chosen centroid.
3. Choose the next centroid with probability proportional to $D(x_i)^2$.
4. Repeat steps 2–3 until K centroids are chosen.
5. Run standard k-means starting from these centroids.

Intuitively, points far from existing centroids have a higher chance of being chosen, ensuring the initial centroids are well-separated.

4.2 Benefits

- Reduces the number of iterations needed for convergence.
- Often avoids extremely poor local minima.
- Simple to implement and widely available in practice (e.g. scikit-learn's `init='k-means++'`).

The extra computational cost of the initialization is usually negligible compared to the cost of the entire clustering run, especially for large datasets.

5 k-medoids: Robust Centroid-based Clustering

While k-means defines centroids as arithmetic means, k-medoids uses actual data points as *representative objects* (medoids) for each cluster [1]. This yields a more robust algorithm in the presence of outliers and when using arbitrary distance measures.

5.1 Objective Function

The objective for k-medoids is similar to Eq. (1), but with c_k restricted to be one of the data points:

$$J_{\text{k-medoids}} = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik}\, d(x_i, m_k), \tag{4}$$

where m_k is the medoid (representative point) of cluster k.

Unlike k-means, k-medoids can use any distance or dissimilarity measure $d(\cdot, \cdot)$, not necessarily Euclidean or metric.

5.2 PAM Algorithm (Partitioning Around Medoids)

One classical algorithm for k-medoids is PAM:

1. Initialize K medoids (e.g. randomly).
2. Assign each point to the nearest medoid.
3. Try to improve the set of medoids by swapping a medoid with a non-medoid point and checking if the total cost decreases.
4. Repeat until no swap reduces the objective.

Because medoids are actual data points, they are less influenced by extreme outliers than means. However, this robustness comes at a computational cost: evaluating all possible swaps can be expensive for large datasets, although faster approximations exist.

5.3 Comparison to k-means

Pros of k-medoids:

- More robust to noise and outliers.
- Works with arbitrary distance matrices (not just vector spaces).

Cons:

- Slower than k-means, especially for large n.
- Implementation is more complex.

In practice, k-means is often preferred for large, numeric datasets; k-medoids is useful when robustness and flexibility of distance are more important than raw speed.

6 Conclusion

Centroid-based clustering offers a simple, intuitive, and powerful framework for grouping data. By representing each cluster with a prototype and assigning points based on distance to that prototype, algorithms like k-means and k-medoids provide fast and interpretable clusterings that scale to real-world problems.

However, these methods are not universally applicable. Their assumptions—globular clusters, meaningful Euclidean geometry, hard assignments—must be checked against the nature of the data. When those assumptions hold approximately, centroid-based clustering is often the first and best tool to try. When they do not, density-based, hierarchical, or spectral methods may offer better alternatives.

Understanding both the strengths and the limitations of centroid-based clustering is essential for using it wisely in practice. With careful preprocessing (especially feature scaling), thoughtful initialization (k-means++), and awareness of outliers (k-medoids), these algorithms can reveal structure in unlabeled data that is both useful and actionable.

References

[1] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2006.

[3] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, 1967, pp. 281–297.